HOLES.M

```
## usage: h = holes(y,p,M)
## Function discovered by Edwin A. Suominen
## Written for Octave (GNU MATLAB alternative)

function [h,yi] = holes(y,p,M)

h = zeros(p-M-1,2);

## Compute inverse of y mod p
[d,yi]= gcd(p,y);
if ( yi(2) < 0 )
  yi = p + yi(2);
elseif
  yi = yi(2);
endif

## Compute column 1 of LUT for this key y:
## holes in ascending order
kk = 0; # Counter for iterating next valid hole value
## For all possible hole values...
for i = 1:p-M-1
  ## Compute prospective hole value (may not be valid)
  h(i,1) = M+1 - rem( i*y-(p-M-1) ,p);
  ## If not valid (if >M), set to flag value
  if ( (h(i,1)>M) | (h(i,1)<1) )
    h(i,:) = zeros(1,2);
  else
    kk++; # Increment valid holes counter
  endif
endfor

## Compute column 2 of LUT for this key y: all possible
## overflowing values, xy mod p > M, in ascending order
kk2 = 1; # Counter for iterating next poss. overflow value
## For M+1 (lowest overflow) to p-1 (highest possible)...
for i = M+1:p-1
  ## Compute input value that would produce each
  ## possible overflowing output
  if ( rem(i*yi,p) <= M ) # If input valid...
    h(kk2,2) = i; # ...assign overflow to LUT.
    kk2++; # Move to next available LUT entry
  endif
endfor

## Sort ascending by values in each column
h = sort(h);
if ( 1+(length(h)-kk) > length(h) )
```

```
  h = 0; # If there are no holes (e.g., for y=1)
else
  h = h(1+(length(h)-kk):length(h),:); # Shrink h to omit flag values
endif

endfunction
```

ENCRYPT.M

```
## usage: z = encrypt (x,y,N,k)
## INPUT: input block(s) x, key y, block length N in bits,
## k offset of modulus from 2^N (p=2^N+k)
## OUTPUT: z = x*y mod (2^N+k), but if
## z >= p, z = ( (z-2^N)*y - (k-1) ) mod (2^N+k )
## Function discovered by Edwin A. Suominen
## Written for Octave (GNU MATLAB alternative)

function z = encrypt(x,y,N,k)

L = length(x); # multiple input blocks can be supplied in a vector
z = zeros(L,1); # initialize output vector

## Enforce k must be odd
if ((k/2)==floor(k/2))
  disp('2^N+k cannot be prime is k is even!');
  return;
endif

## Define set order (M) and modulus (p)
M = 2^N; p = M+k;

## Compute LUT of holes in ascending order
## for this key y
h = holes(y,p,M);
Nh = rows(h);

## Basic modulo multipication operation
## Do as array to speed things up
z = rem(x.*y,p);

## For each element in vector...
for i = 1:L

  ## Inventive exception handling
  if (z(i) > M)
    ## Map overflowing value to corresponding hole value in LUT
    ## If there are no holes (h=1 scalar), this code will not
    ## be called because z will always be <= M.
    c = 1:Nh; c = c'; # 1,2 ... (# of valid holes)
    c = c .* ( (z(i)*ones(Nh,1))==h(:,2) ); # Zeros with index of match
    ## z = hole from LUT entry having matching overflow value
    z(i) = h(max(c),1);
  endif

endfor
endfunction
```

DECRYPT.M

```
## usage: x = decrypt (z,y,N,k)
## INPUT: encoded block(s) z, key y, block length N in bits,
## k offset of modulus from 2^N (p=2^N+k)
## OUTPUT: x = z*y^-1 mod (2^N+k), but if
## z = h, where h = ( ((1:k)*y - (k-1) ) mod (2^N+k )
## then z = M+a, where
## a = y^-1 * (2*M+(2+p-h)) mod (2^N+k)
## Function discovered by Edwin A. Suominen
## Written for Octave (GNU MATLAB alternative)

function x = decrypt (z,y,N,k)

L = length(z);  # multiple input blocks can be supplied in a vector

## Enforce k must be odd
if ((k/2)==floor(k/2))
  disp('2^N+k cannot be prime is k is even!');
  return;
endif

## Define set order (M) and modulus (p)
M = 2^N; p = M+k;

## Compute LUT of holes in ascending order
## for this key y
[h,y] = holes(y,p,M); # With two args out, returns y^-1
if ( size(h)==1 )
  Nh = 0; # Account for special case of no holes
else
  Nh = rows(h);
endif

## Done with encryption key y, now y is modulo inverse of orig. y

## For all encrypted blocks (values)...
for i = 1:L

  if Nh>0
    ## If z(i) has been mapped to a hole, restore to overflowing value
    ## For all possible hole values given this key
    for j = 1:Nh
      ## If matches a hole value, remap back
      if (z(i)==h(j,1)), z(i) = h(j,2); endif
    endfor
  endif

endfor
```

```
## Now invert remapped values in vector
## Restored overflowing values will be decrypted properly.
## Do as array to speed things up
x = rem(z.*y,p);      # y = y^1 at this point

endfunction
```

HOLETEST.M

```
## HOLETEST.M
## Written for Octave (GNU MATLAB alternative)

# Np = 1; M = 128; p = M + 3;
Np = 1; M = 512; p = M + 9;

% Try all column (key) values in {1,2,...M}
for j = 2:M,

% Get hole values with brute-force lookup method
x1 = holes1(j,p,M);

% Get hole values using formula discovered by Ed Suominen
x2 = holes2(j,p,M);

disp('');
disp(['j=',num2str(j)]);
disp('  -x1-  -x2-');
disp([x1 x2]);

% Compare
err(j) = sum(abs(x1-x2));
disp(['Sum of absolute differences = ',num2str(err(j))]);

endfor
```

HOLES1.M

```
function h = holes1(y,p,M);

% h = holes1(y,p,N);
% Finds "holes" - skipped values of set {0,1}^N in result
% of x*y mod p. Variable length result with only holes.

% Number of values in set S:{0,1}^N
% M = 2^N;

s = 1:M;   % Working array of values in set S

% Zero out values in set that occur ("non-holes")
  for i = 1:M
    j = rem(i*y,p);  % xy mod p
    % Zero out if not a hole
    if j<=M, s(j) = 0; end
  endfor

% Sort decending to get holes first
h = -sort(-s);
% Trim off zeros (non-holes)
Nnz = sum(h>0); h = h(1:Nnz)';

endfunction
```

HOLES2.M

```
function h = holes2(y,p,M);

% h = holes2(y,p,M)
% Finds "holes" - skipped values of set {0,1}^N in result
% of x*y mod p.
% Uses equation discovered by Edwin A. Suominen

% Number of values in set S:{0,1}^N
% M = 2^N;

k = p-(M+1);

% For vector inputs...
for i=1:length(y)

  for j=1:k,

    ## Input values between M+1 and p will of necessity
    ## be mapped to holes (values not produced by inputs
    ## from set {1,2,...M} because xy mod p is a bijection
    ## (See HAC 1.8 Definition)
    ## h(j,i) = rem( (M+j)*y ,p);
    ## Equation above is simple but doesn't work when
    ## M < xy < p (which happens rarely, but it happens).

    h(j,i) = M+1 - rem(j*y(i)-k,p);
  endfor

endfor

% Map negs. to 0, Sort decending to match formats
Nok = sum(h<=M); h = sort(h); h = h(1:Nok);
if Nok==0, h = []; endif
h = h.*(h>0);
h = -sort(-h);
% Trim off zeros (non-holes)
Nnz = sum(h>0); h = h(1:Nnz);

endfunction
```

```
octave:56> date
ans = [redacted]
octave:57> clock
ans =

  2000.0000          9.0000        30.0000
12.0000    35.0000    32.0890

octave:58> type holes1
holes1  is  the  user-defined  function
defined from: /1093-2/holes1.m

function h = holes1(y,p,M);

% h = holes1(y,p,N);
% CONFIDENTIAL AND PROPRIETARY
% Edwin A. Suominen
% 091600 - Initial writing
% Finds "holes" - skipped values of set
{0,1}^N in result
% of x*y mod p. Variable length result
with only holes.

% Number of values in set S:{0,1}^N
% M = 2^N;

s = 1:M;    % Working array of values in
set S

% Zero out values in set that occur
("non-holes")
  for i = 1:M
    j = rem(i*y,p);  % xy mod p
    % Zero out if not a hole
    if j<=M, s(j) = 0; end
  endfor


% Sort decending to get holes first
h = -sort(-s);
% Trim off zeros (non-holes)
Nnz = sum(h>0); h = h(1:Nnz)';

endfunctionoctave:59> type holes2
holes2  is  the  user-defined  function
defined from: /1093-2/holes2.m

function h = holes2(y,p,M);

% h = holes2(y,p,M)
[redacted]
% Finds "holes" - skipped values of set
{0,1}^N in result
% of x*y mod p.
% Uses equation discovered by EAS [redacted]

% Number of values in set S:{0,1}^N
% M = 2^N;

k = p-(M+1);

% For vector inputs...
for i=1:length(y)

  for j=1:k,

    ## Input values between M+1 and p
will of necessity
    ## be mapped to holes (values not
produced by inputs
    ## from set {1,2,...M} because xy mod
p is a bijection
    ## (See HAC 1.8 Definition)
    h(j,i) = rem( (M+j)*y ,p);

    ## The simple equation above is
substituted for the one below
    ## h(j,i) = M+1 - rem(j*y(i)-k,p);
  endfor

endfor

% Map negs. to 0, Sort decending to match
formats
Nok = sum(h<=M); h = sort(h); h =
h(1:Nok);
if Nok==0, h = []; endif
h = h.*(h>0);
h = -sort(-h);
% Trim off zeros (non-holes)
Nnz = sum(h>0); h = h(1:Nnz);

endfunctionoctave:60> type holetest
holetest is the script file: /1093-
2/holetest.m

## HOLETEST.M
## This file is CONFIDENTIAL AND
PROPRIETARY.
[redacted]
## Written for Octave (GNU MATLAB
alternative)
## REVISION [redacted]
```

[redacted]

```
Np = 1; M = 512; p = M + 9;

%  Try  all  column  (key)  values  in
{1,2,...M}
for j = 2:M,

% Get hole values with brute-force lookup
method
x1 = holes1(j,p,M);

%  Get  hole  values  using  formula
discovered [redacted]
% by Ed Suominen
x2 = holes2(j,p,M);

disp('');
disp(['j=',num2str(j)]);
disp('  -x1-  -x2-');
disp([x1 x2]);


% Compare
err(j) = sum(abs(x1-x2));
disp(['Sum  of  absolute  differences  =
',num2str(err(j))]);

endforoctave:61> who

*** currently compiled functions:

clock   date    holes1  holes2

octave:62> holetest

j=2
  -x1-  -x2-
  511   511
  509   509
  507   507
  505   505
Sum of absolute differences = 0

j=3
  -x1-  -x2-
  512   512
  509   509
  506   506
  503   503
  500   500
  497   497
Sum of absolute differences = 0

j=4
  -x1-  -x2-
  509   509
  505   505
  501   501
  497   497
  493   493
  489   489
Sum of absolute differences = 0

j=5
  -x1-  -x2-
  511   511
  506   506
  501   501
  496   496
  491   491
  486   486
  481   481
Sum of absolute differences = 0

j=6
  -x1-  -x2-
  509   509
  503   503
  497   497
  491   491
  485   485
  479   479
  473   473
Sum of absolute differences = 0

j=7
  -x1-  -x2-
  507   507
  500   500
  493   493
  486   486
  479   479
  472   472
  465   465
Sum of absolute differences = 0

j=8
  -x1-  -x2-
  505   505
  497   497
  489   489
  481   481
  473   473
  465   465
  457   457
Sum of absolute differences = 0

j=9
  -x1-  -x2-
  512   512
  503   503
  494   494
  485   485
  476   476
```

```
  467  467
  458  458
  449  449
Sum of absolute differences = 0

j=10
  -x1-  -x2-
  511  511
  501  501
  491  491
  481  481
  471  471
  461  461
  451  451
  441  441
Sum of absolute differences = 0

j=11
  -x1-  -x2-
  510  510
  499  499
  488  488
  477  477
  466  466
  455  455
  444  444
  433  433
Sum of absolute differences = 0

j=12
  -x1-  -x2-
  509  509
  497  497
  485  485
  473  473
  461  461
  449  449
  437  437
  425  425
Sum of absolute differences = 0

j=13
  -x1-  -x2-
  508  508
  495  495
  482  482
  469  469
  456  456
  443  443
  430  430
  417  417
Sum of absolute differences = 0

j=14
  -x1-  -x2-
  507  507
  493  493
  479  479
  465  465
  451  451
  437  437
  423  423
  409  409
Sum of absolute differences = 0

j=15
  -x1-  -x2-
  506  506
  491  491
  476  476
  461  461
  446  446
  431  431
  416  416
  401  401
Sum of absolute differences = 0

j=16
  -x1-  -x2-
  505  505
  489  489
  473  473
  457  457
  441  441
  425  425
  409  409
  393  393
Sum of absolute differences = 0

j=17
  -x1-  -x2-
  504  504
  487  487
  470  470
  453  453
  436  436
  419  419
  402  402
  385  385
Sum of absolute differences = 0

j=18
  -x1-  -x2-
  503  503
  485  485
  467  467
  449  449
  431  431
  413  413
  395  395
  377  377
Sum of absolute differences = 0

j=19
  -x1-  -x2-
  502  502
```

```
  162  162
  108  108
   54   54
Sum of absolute differences = 0

j=468
  -x1-  -x2-
  424  424
  371  371
  318  318
  265  265
  212  212
  159  159
  106  106
   53   53
Sum of absolute differences = 0

j=469
  -x1-  -x2-
  416  416
  364  364
  312  312
  260  260
  208  208
  156  156
  104  104
   52   52
Sum of absolute differences = 0

j=470
  -x1-  -x2-
  408  408
  357  357
  306  306
  255  255
  204  204
  153  153
  102  102
   51   51
Sum of absolute differences = 0

j=471
  -x1-  -x2-
  400  400
  350  350
  300  300
  250  250
  200  200
  150  150
  100  100
   50   50
Sum of absolute differences = 0

j=472
  -x1-  -x2-
  392  392
  343  343
  294  294
  245  245
  196  196
  147  147
   98   98
   49   49
Sum of absolute differences = 0

j=473
  -x1-  -x2-
  384  384
  336  336
  288  288
  240  240
  192  192
  144  144
   96   96
   48   48
Sum of absolute differences = 0

j=474
  -x1-  -x2-
  376  376
  329  329
  282  282
  235  235
  188  188
  141  141
   94   94
   47   47
Sum of absolute differences = 0

j=475
  -x1-  -x2-
  368  368
  322  322
  276  276
  230  230
  184  184
  138  138
   92   92
   46   46
Sum of absolute differences = 0

j=476
  -x1-  -x2-
  360  360
  315  315
  270  270
  225  225
  180  180
  135  135
   90   90
   45   45
Sum of absolute differences = 0

j=477
  -x1-  -x2-
  352  352
```

```
   308  308
   264  264
   220  220
   176  176
   132  132
    88   88
    44   44
 Sum of absolute differences = 0

 j=478
  -x1-  -x2-
   344  344
   301  301
   258  258
   215  215
   172  172
   129  129
    86   86
    43   43
 Sum of absolute differences = 0

 j=479
  -x1-  -x2-
   336  336
   294  294
   252  252
   210  210
   168  168
   126  126
    84   84
    42   42
 Sum of absolute differences = 0

 j=480
  -x1-  -x2-
   328  328
   287  287
   246  246
   205  205
   164  164
   123  123
    82   82
    41   41
 Sum of absolute differences = 0

 j=481
  -x1-  -x2-
   320  320
   280  280
   240  240
   200  200
   160  160
   120  120
    80   80
    40   40
 Sum of absolute differences = 0

 j=482
  -x1-  -x2-
   312  312
   273  273
   234  234
   195  195
   156  156
   117  117
    78   78
    39   39
 Sum of absolute differences = 0

 j=483
  -x1-  -x2-
   304  304
   266  266
   228  228
   190  190
   152  152
   114  114
    76   76
    38   38
 Sum of absolute differences = 0

 j=484
  -x1-  -x2-
   296  296
   259  259
   222  222
   185  185
   148  148
   111  111
    74   74
    37   37
 Sum of absolute differences = 0

 j=485
  -x1-  -x2-
   288  288
   252  252
   216  216
   180  180
   144  144
   108  108
    72   72
    36   36
 Sum of absolute differences = 0

 j=486
  -x1-  -x2-
   280  280
   245  245
   210  210
   175  175
   140  140
   105  105
    70   70
    35   35
 Sum of absolute differences = 0
```

```
j=487
  -x1-  -x2-
  272   272
  238   238
  204   204
  170   170
  136   136
  102   102
   68    68
   34    34
Sum of absolute differences = 0

j=488
  -x1-  -x2-
  264   264
  231   231
  198   198
  165   165
  132   132
   99    99
   66    66
   33    33
Sum of absolute differences = 0

j=489
  -x1-  -x2-
  256   256
  224   224
  192   192
  160   160
  128   128
   96    96
   64    64
   32    32
Sum of absolute differences = 0

j=490
  -x1-  -x2-
  248   248
  217   217
  186   186
  155   155
  124   124
   93    93
   62    62
   31    31
Sum of absolute differences = 0

j=491
  -x1-  -x2-
  240   240
  210   210
  180   180
  150   150
  120   120
   90    90
   60    60
   30    30
Sum of absolute differences = 0

j=492
  -x1-  -x2-
  232   232
  203   203
  174   174
  145   145
  116   116
   87    87
   58    58
   29    29
Sum of absolute differences = 0

j=493
  -x1-  -x2-
  224   224
  196   196
  168   168
  140   140
  112   112
   84    84
   56    56
   28    28
Sum of absolute differences = 0

j=494
  -x1-  -x2-
  216   216
  189   189
  162   162
  135   135
  108   108
   81    81
   54    54
   27    27
Sum of absolute differences = 0

j=495
  -x1-  -x2-
  208   208
  182   182
  156   156
  130   130
  104   104
   78    78
   52    52
   26    26
Sum of absolute differences = 0

j=496
  -x1-  -x2-
  200   200
  175   175
  150   150
  125   125
  100   100
```

```
   75   75
   50   50
   25   25
 Sum of absolute differences = 0

 j=497
   -x1-  -x2-
  192  192
  168  168
  144  144
  120  120
   96   96
   72   72
   48   48
   24   24
 Sum of absolute differences = 0

 j=498
   -x1-  -x2-
  184  184
  161  161
  138  138
  115  115
   92   92
   69   69
   46   46
   23   23
 Sum of absolute differences = 0

 j=499
   -x1-  -x2-
  176  176
  154  154
  132  132
  110  110
   88   88
   66   66
   44   44
   22   22
 Sum of absolute differences = 0

 j=500
   -x1-  -x2-
  168  168
  147  147
  126  126
  105  105
   84   84
   63   63
   42   42
   21   21
 Sum of absolute differences = 0

 j=501
   -x1-  -x2-
  160  160
  140  140
  120  120
  100  100
   80   80
   60   60
   40   40
   20   20
 Sum of absolute differences = 0

 j=502
   -x1-  -x2-
  152  152
  133  133
  114  114
   95   95
   76   76
   57   57
   38   38
   19   19
 Sum of absolute differences = 0

 j=503
   -x1-  -x2-
  144  144
  126  126
  108  108
   90   90
   72   72
   54   54
   36   36
   18   18
 Sum of absolute differences = 0

 j=504
   -x1-  -x2-
  136  136
  119  119
  102  102
   85   85
   68   68
   51   51
   34   34
   17   17
 Sum of absolute differences = 0

 j=505
   -x1-  -x2-
  128  128
  112  112
   96   96
   80   80
   64   64
   48   48
   32   32
   16   16
 Sum of absolute differences = 0

 j=506
   -x1-  -x2-
  120  120
```

```
 105  105
  90   90
  75   75
  60   60
  45   45
  30   30
  15   15
Sum of absolute differences = 0

j=507
 -x1-  -x2-
 112  112
  98   98
  84   84
  70   70
  56   56
  42   42
  28   28
  14   14
Sum of absolute differences = 0

j=508
 -x1-  -x2-
 104  104
  91   91
  78   78
  65   65
  52   52
  39   39
  26   26
  13   13
Sum of absolute differences = 0

j=509
 -x1-  -x2-
 96  96
 84  84
 72  72
 60  60
 48  48
 36  36
 24  24
 12  12
Sum of absolute differences = 0

j=510
 -x1-  -x2-
 88  88
 77  77
 66  66
 55  55
 44  44
 33  33
 22  22
 11  11
Sum of absolute differences = 0

j=511
 -x1-  -x2-
 80  80
 70  70
 60  60
 50  50
 40  40
 30  30
 20  20
 10  10
Sum of absolute differences = 0

j=512
 -x1-  -x2-
 72  72
 63  63
 54  54
 45  45
 36  36
 27  27
 18  18
  9   9
Sum of absolute differences = 0
octave:63> who

*** currently compiled functions:

clock        columns    date          holes1
holes2    num2str  rem      rows

*** local user variables:

M    Np    err  j    p    x1   x2

octave:64> size(M)
ans =

  1  1

octave:65> size(x1)
ans =

  8  1

octave:66> size(err)
ans =

  512    1

octave:67> max(abs(err))
ans = 0
octave:68> 'Simple hole finding function works!'
ans = Simple hole finding function works!
octave:69> clock
ans =
[redacted]
octave:70> diary off
```

HOLES3.M

```
function [k,h] = holes3(y,p,M);

% h = holes3(y,p,M)
% CONFIDENTIAL AND PROPRIETARY
% Edwin A. Suominen
% Finds "holes" - skipped values of set {0,1}^N in result
% of x*y mod p.
% Uses equation discovered by EAS 9/16/00

% Number of values in set S:{0,1}^N
% M = 2^N;

k = p-(M+1);

% For vector inputs...
for i=1:length(y)

  for j=1:k,

    ## Input values between M+1 and p will of necessity
    ## be mapped to holes (values not produced by inputs
    ## from set {1,2,...M} because xy mod p is a bijection
    ## (See HAC 1.8 Definition)
    ## h(j,i) = rem( (M+j)*y ,p);
    ## Equation above is simple but doesn't work when
    ## M < xy < p (which happens rarely, but it happens).

    h(j,i) = M+1 - rem(j*y(i)-k,p);
  endfor

endfor




if (nargout>=2)
  k = 1:k; k=k';
endif

endfunction
```

TEST3.M

TESTS EACH INPUT FOR ALL KEYS IN SPACE

```
## TEST3.M
## Block size is 10 bits. Input is taken from set Z:{1,2,...1024}
## Because of EAS-invented "pseudogroup" operation, output also
## falls in set Z.
## Keys are also taken from set Z - any set element is OK.

## This test proves the following:
## (1) Output set is same as input set Z.
## (2) Each input value has a unique output value, for a given
## key value.
## (3) The output value from "encrypt.m" can be converted back to
## the input value with "decrypt.m," given the key value.
## (4) For a given input value, each key value produces a unique
## output value.
## Written for Octave (GNU MATLAB alternative)

## No paging - want current screen output
page_screen_output=0;

## Set values defining set and underlying group order
N = 10; M = 2^N; # M = 1024
k = 7; p = M+k; # p = 1031 (prime)

## Create empty matrix of output values
A = zeros(M);

## Define vector with elements of set Z
v = linspace(1,M,M);

## Create string matrix of '-' neutral values for test condition codes
cc = ['-RESULTS- '; ' key: 1234']; # Header
## for each key value...
for i = 1:M
  ## insert key value before neutrals
  ccr = [num2str(i),': ----'];
  ## Leading zeros to make columns line up
  if i<10, ccr = ['0' ccr]; endif
  if i<100, ccr = ['0' ccr]; endif
  if i<1000, ccr = ['0' ccr]; endif
  cc(i+2,:) = ccr;
endfor

############ PART ONE OF TWO #############
disp(['Tests 1-3, for each key value in set 1,2,...',num2str(M)]);
disp('-----------------------------------------------------');
```

```
## For all possible key values in Z...
for i = 1:M

  ## Show progress
  disp(['Encrypting and decrypting with key y=',num2str(i),'...']);

  ## Set key value for this iteration
  y = v(i);

  ## Encrypt all possible input values in set Z with key
  b = encrypt(v,y,N,k);
  A(:,i) = b'; # Add this output vector to output matrix

  ## Test for conditions (1),(2) now
  b = sort(b); # Sort ascending

  disp(['Output set: min=',num2str(min(b)),', max=',num2str(max(b))]);


  ##### Test Condition (1) #####
  if ( max(b)==M )
     disp('Output set is same as input set.');
    cc(i+2,7) = '+';
  else
    disp('PROBLEM: Output set larger or smaller than input set!');
    cc(i+2,7) = 'o';
  endif


  ##### Test Condition (2) #####
  ## Each input value should have a unique output value, for a given
  ## key value.
  b = diff(b); # Get differentials between sorted elements
  if ( min(b)==1 & max(b)==1 )
    disp('All elements in output set are unique.');
    cc(i+2,8) = '+';
  else
    disp('PROBLEM: skipped or duplicated element(s) in output set!');
    cc(i+2,8) = 'o';
  endif


  ##### Decrypt output values for this key #####
  b = decrypt(A(:,i),y,N,k)';


  ##### Test Condition (3) #####
```

```
  ## Get differentials between plaintext-encrypted-decrypted (b) and plaintext (v)
  b = b - v; # Should be all zeros if test passes
  if ( (max(abs(b))==0) )
    disp('All elements in input set encrypt and decrypt with key and inverse.');
    cc(i+2,9) = '+';
  else
    disp('PROBLEM: One or more elements do not match in encryption/decryption!');
    cc(i+2,9) = 'o';
  endif

  disp('');

endfor

############ PART TWO OF TWO #############
disp(['Test 4, for each input value in set 1,2,...',num2str(M)]);
disp('----------------------------------------------------------------');

##### Test Condition (4) #####

## For all possible input values in Z, working with full matrix of outputs
for i = 1:M

  ## Show progress
  disp(['Analyzing outputs for input x=',num2str(i),' with all keys in set...']);

  ## For a given input value, each key value should produce a unique output value.
  b = diff(sort(A(i,:))); # Get differentials between sorted elements
  if ( min(b)==1 & max(b)==1 )
    disp('All elements in output set are unique.');
    cc(i+2,10) = '+';
  else
    disp('Skipped or duplicated element(s) in output set.');
    cc(i+2,10) = 'o';
  endif

  disp('');

endfor

## Display test results
disp(cc)
```

RESULTS OF TEST3.M

```
octave:14> date
ans = [redacted]
octave:15> clock
ans =

[redacted]

octave:16> type test3
test3 is the script file: /1093-2/test3.m

## TEST3.M
## Block size is 10 bits. Input is taken from set
Z:{1,2,...1024}
## Because of EAS-invented "pseudogroup" operation,
output also
## falls in set Z.
## Keys are also taken from set Z - any set element is
OK.

## This test proves the following:
## (1) Output set is same as input set Z.
## (2) Each input value has a unique output value, for a
given
## key value.
## (3) The output value from "encrypt.m" can be converted
back to
## the input value with "decrypt.m," given the key value.
## (4) For a given input value, each key value produces a
unique
## output value.

[redacted]
## Written for Octave (GNU MATLAB alternative)
[redacted]

## No paging - want current screen output
page_screen_output=0;

## Set values defining set and underlying group order
N = 10; M = 2^N; # M = 1024
k = 7; p = M+k; # p = 1031 (prime)

## Create empty matrix of output values
A = zeros(M);

## Define vector with elements of set Z
v = linspace(1,M,M);

## Create string matrix of '-' neutral values for test
condition codes
cc = ['-RESULTS- '; ' key: 1234']; # Header
## for each key value...
for i = 1:M
  ## insert key value before neutrals
  ccr = [num2str(i),': ----'];
  ## Leading zeros to make columns line up
  if i<10, ccr = ['0' ccr]; endif
  if i<100, ccr = ['0' ccr]; endif
  if i<1000, ccr = ['0' ccr]; endif
  cc(i+2,:) = ccr;
endfor
```

```
############ PART ONE OF TWO ##############
disp(['Tests 1-3, for each key value in s
1,2,...',num2str(M)]);
disp('---------------------------------------------------
----');

## For all possible key values in Z...
for i = 1:M

  ## Show progress
  disp(['Encrypting and decrypting with k
y=',num2str(i),'...']);

  ## Set key value for this iteration
  y = v(i);

  ## Encrypt all possible input values in set Z with key
  b = encrypt(v,y,N,k);
  A(:,i) = b'; # Add this output vector to output matrix

  ## Test for conditions (1),(2) now
  b = sort(b); # Sort ascending

  disp(['Output set: min=',num2str(min(b)),
max=',num2str(max(b))]);

  ##### Test Condition (1) #####
  if ( max(b)==M )
     disp('Output set is same as input set.');
    cc(i+2,7) = '+';
  else
    disp('PROBLEM: Output set larger or smaller thi
input set!');
    cc(i+2,7) = 'o';
  endif


  ##### Test Condition (2) #####
  ## Each input value should have a unique output value
for a given
  ## key value.
  b = diff(b); # Get differentials between sorte
elements
  if ( min(b)==1 & max(b)==1 )
    disp('All elements in output set are unique.');
    cc(i+2,8) = '+';
  else
    disp('PROBLEM: skipped or duplicated element(s) i
output set!');
    cc(i+2,8) = 'o';
  endif


  ##### Decrypt output values for this key #####
  b = decrypt(A(:,i),y,N,k)';


  ##### Test Condition (3) #####
  ## Get differentials between plaintext-encrypted
decrypted (b) and plaintext (v)
  b = b - v; # Should be all zeros if test passes
  if ( (max(abs(b))==0) )
    disp('All elements in input set encrypt and decryp
with key and inverse.');
    cc(i+2,9) = '+';
```

```
  else
    disp('PROBLEM: One or more elements do not match in encryption/decryption!');
    cc(i+2,9) = 'o';
  endif

  disp('');

endfor


############ PART TWO OF TWO ##############
disp(['Test 4, for each input value in set 1,2,...',num2str(M)]);
disp('--------------------------------------------------------');

##### Test Condition (4) #####

## For all possible input values in Z, working with full matrix of outputs
for i = 1:M

  ## Show progress
  disp(['Analyzing outputs for input x=',num2str(i),' with all keys in set...']);

  ## For a given input value, each key value should produce a unique output value.
  b = diff(sort(A(i,:))); # Get differentials between sorted elements
  if ( min(b)==1 & max(b)==1 )
    disp('All elements in output set are unique.');
    cc(i+2,10) = '+';
  else
    disp('Skipped or duplicated element(s) in output set.');
    cc(i+2,10) = 'o';
  endif

  disp('');

endfor

## Display test results
disp(cc)
octave:18> test3
Tests 1-3, for each key value in set 1,2,...1024
------------------------------------------------------
Encrypting and decrypting with key y=1...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key and inverse.

Encrypting and decrypting with key y=2...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key and inverse.

Encrypting and decrypting with key y=3...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key and inverse.

Encrypting and decrypting with key y=4...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with k
and inverse.

Encrypting and decrypting with key y=5...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with k
and inverse.

Encrypting and decrypting with key y=6...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with k
and inverse.

Encrypting and decrypting with key y=7...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with k
and inverse.

Encrypting and decrypting with key y=8...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.

Encrypting and decrypting with key y=9...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.

Encrypting and decrypting with key y=10...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.

Encrypting and decrypting with key y=11...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.

Encrypting and decrypting with key y=12...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.

Encrypting and decrypting with key y=13...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.

Encrypting and decrypting with key y=14...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.
```

```
Encrypting and decrypting with key y=15...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=16...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=17...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=18...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=19...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=20...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=21...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=22...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=23...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=24...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=25...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with k
and inverse.

Encrypting and decrypting with key y=26...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with k
and inverse.

Encrypting and decrypting with key y=27...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with k
and inverse.

Encrypting and decrypting with key y=28...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.

Encrypting and decrypting with key y=29...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.

Encrypting and decrypting with key y=30...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.

Encrypting and decrypting with key y=31...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.

Encrypting and decrypting with key y=32...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.

Encrypting and decrypting with key y=33...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.

Encrypting and decrypting with key y=34...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with ke
and inverse.

Encrypting and decrypting with key y=35...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
```

```
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=1014...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=1015...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=1016...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=1017...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=1018...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=1019...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=1020...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=1021...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=1022...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with key
and inverse.

Encrypting and decrypting with key y=1023...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with k
and inverse.

Encrypting and decrypting with key y=1024...
Output set: min=1, max=1024
Output set is same as input set.
All elements in output set are unique.
All elements in input set encrypt and decrypt with k
and inverse.

Test 4, for each input value in set 1,2,...1024
------------------------------------------------------
Analyzing outputs for input x=1 with all keys in set...
All elements in output set are unique.

Analyzing outputs for input x=2 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=3 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=4 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=5 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=6 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=7 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=8 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=9 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=10 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=11 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=12 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=13 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=14 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=15 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=16 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=17 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=18 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=19 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=20 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=21 with all keys in set...
```

```
Analyzing outputs for input x=995 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=996 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=997 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=998 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=999 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1000 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1001 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1002 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1003 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1004 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1005 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1006 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1007 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1008 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1009 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1010 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1011 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1012 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1013 with all keys in set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1014 with all keys set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1015 with all keys set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1016 with all keys set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1017 with all keys set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1018 with all keys set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1019 with all keys set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1020 with all keys set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1021 with all keys set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1022 with all keys set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1023 with all keys set...
Skipped or duplicated element(s) in output set.

Analyzing outputs for input x=1024 with all keys set...
Skipped or duplicated element(s) in output set.

-RESULTS-
 key: 1234
0001: ++++
0002: +++o
0003: +++o
0004: +++o
0005: +++o
0006: +++o
0007: +++o
0008: +++o
0009: +++o
0010: +++o
0011: +++o
0012: +++o
0013: +++o
0014: +++o
0015: +++o
0016: +++o
0017: +++o
0018: +++o
0019: +++o
0020: +++o
0021: +++o
0022: +++o
0023: +++o
0024: +++o
0025: +++o
0026: +++o
0027: +++o
```

0028: +++o
0029: +++o
0030: +++o
0031: +++o
0032: +++o
0033: +++o
0034: +++o
0035: +++o
0036: +++o
0037: +++o
0038: +++o
0039: +++o
0040: +++o
0041: +++o
0042: +++o
0043: +++o
0044: +++o
0045: +++o
0046: +++o
0047: +++o
0048: +++o
0049: +++o
0050: +++o
0051: +++o
0052: +++o
0053: +++o
0054: +++o
0055: +++o
0056: +++o
0057: +++o
0058: +++o
0059: +++o
0060: +++o
0061: +++o
0062: +++o
0063: +++o
0064: +++o
0065: +++o
0066: +++o
0067: +++o
0068: +++o
0069: +++o
0070: +++o
0071: +++o
0072: +++o
0073: +++o
0074: +++o
0075: +++o
0076: +++o
0077: +++o
0078: +++o
0079: +++o
0080: +++o
0081: +++o
0082: +++o
0083: +++o
0084: +++o
0085: +++o
0086: +++o
0087: +++o
0088: +++o
0089: +++o
0090: +++o
0091: +++o
0092: +++o
0093: +++o
0094: +++o
0095: +++o
0096: +++o
0097: +++o
0098: +++o
0099: +++o
0100: +++o
0101: +++o
0102: +++o
0103: +++o
0104: +++o
0105: +++o
0106: +++o
0107: +++o
0108: +++o
0109: +++o
0110: +++o
0111: +++o
0112: +++o
0113: +++o
0114: +++o
0115: +++o
0116: +++o
0117: +++o
0118: +++o
0119: +++o
0120: +++o
0121: +++o
0122: +++o
0123: +++o
0124: +++o
0125: +++o
0126: +++o
0127: +++o
0128: +++o
0129: +++o
0130: +++o
0131: +++o
0132: +++o
0133: +++o
0134: +++o
0135: +++o
0136: +++o
0137: +++o
0138: +++o
0139: +++o
0140: +++o
0141: +++o
0142: +++o
0143: +++o
0144: +++o
0145: +++o
0146: +++o
0147: +++o
0148: +++o
0149: +++o
0150: +++o
0151: +++o
0152: +++o
0153: +++o
0154: +++o
0155: +++o
0156: +++o
0157: +++o
0158: +++o
0159: +++o
0160: +++o
0161: +++o
0162: +++o
0163: +++o
0164: +++o
0165: +++o
0166: +++o
0167: +++o
0168: +++o
0169: +++o
0170: +++o
0171: +++o
0172: +++o
0173: +++o

0174: +++o
0175: +++o
0176: +++o
0177: +++o
0178: +++o
0179: +++o
0180: +++o
0181: +++o
0182: +++o
0183: +++o
0184: +++o
0185: +++o
0186: +++o
0187: +++o
0188: +++o
0189: +++o
0190: +++o
0191: +++o
0192: +++o
0193: +++o
0194: +++o
0195: +++o
0196: +++o
0197: +++o
0198: +++o
0199: +++o
0200: +++o
0201: +++o
0202: +++o
0203: +++o
0204: +++o
0205: +++o
0206: +++o
0207: +++o
0208: +++o
0209: +++o
0210: +++o
0211: +++o
0212: +++o
0213: +++o
0214: +++o
0215: +++o
0216: +++o
0217: +++o
0218: +++o
0219: +++o
0220: +++o
0221: +++o
0222: +++o
0223: +++o
0224: +++o
0225: +++o
0226: +++o
0227: +++o
0228: +++o
0229: +++o
0230: +++o
0231: +++o
0232: +++o
0233: +++o
0234: +++o
0235: +++o
0236: +++o
0237: +++o
0238: +++o
0239: +++o
0240: +++o
0241: +++o
0242: +++o
0243: +++o
0244: +++o
0245: +++o
0246: +++o
0247: +++o
0248: +++o
0249: +++o
0250: +++o
0251: +++o
0252: +++o
0253: +++o
0254: +++o
0255: +++o
0256: +++o
0257: +++o
0258: +++o
0259: +++o
0260: +++o
0261: +++o
0262: +++o
0263: +++o
0264: +++o
0265: +++o
0266: +++o
0267: +++o
0268: +++o
0269: +++o
0270: +++o
0271: +++o
0272: +++o
0273: +++o
0274: +++o
0275: +++o
0276: +++o
0277: +++o
0278: +++o
0279: +++o
0280: +++o
0281: +++o
0282: +++o
0283: +++o
0284: +++o
0285: +++o
0286: +++o
0287: +++o
0288: +++o
0289: +++o
0290: +++o
0291: +++o
0292: +++o
0293: +++o
0294: +++o
0295: +++o
0296: +++o
0297: +++o
0298: +++o
0299: +++o
0300: +++o
0301: +++o
0302: +++o
0303: +++o
0304: +++o
0305: +++o
0306: +++o
0307: +++o
0308: +++o
0309: +++o
0310: +++o
0311: +++o
0312: +++o
0313: +++o
0314: +++o
0315: +++o
0316: +++o
0317: +++o
0318: +++o
0319: +++o

0320: +++o
0321: +++o
0322: +++o
0323: +++o
0324: +++o
0325: +++o
0326: +++o
0327: +++o
0328: +++o
0329: +++o
0330: +++o
0331: +++o
0332: +++o
0333: +++o
0334: +++o
0335: +++o
0336: +++o
0337: +++o
0338: +++o
0339: +++o
0340: +++o
0341: +++o
0342: +++o
0343: +++o
0344: +++o
0345: +++o
0346: +++o
0347: +++o
0348: +++o
0349: +++o
0350: +++o
0351: +++o
0352: +++o
0353: +++o
0354: +++o
0355: +++o
0356: +++o
0357: +++o
0358: +++o
0359: +++o
0360: +++o
0361: +++o
0362: +++o
0363: +++o
0364: +++o
0365: +++o
0366: +++o
0367: +++o
0368: +++o
0369: +++o
0370: +++o
0371: +++o
0372: +++o
0373: +++o
0374: +++o
0375: +++o
0376: +++o
0377: +++o
0378: +++o
0379: +++o
0380: +++o
0381: +++o
0382: +++o
0383: +++o
0384: +++o
0385: +++o
0386: +++o
0387: +++o
0388: +++o
0389: +++o
0390: +++o
0391: +++o
0392: +++o
0393: +++o
0394: +++o
0395: +++o
0396: +++o
0397: +++o
0398: +++o
0399: +++o
0400: +++o
0401: +++o
0402: +++o
0403: +++o
0404: +++o
0405: +++o
0406: +++o
0407: +++o
0408: +++o
0409: +++o
0410: +++o
0411: +++o
0412: +++o
0413: +++o
0414: +++o
0415: +++o
0416: +++o
0417: +++o
0418: +++o
0419: +++o
0420: +++o
0421: +++o
0422: +++o
0423: +++o
0424: +++o
0425: +++o
0426: +++o
0427: +++o
0428: +++o
0429: +++o
0430: +++o
0431: +++o
0432: +++o
0433: +++o
0434: +++o
0435: +++o
0436: +++o
0437: +++o
0438: +++o
0439: +++o
0440: +++o
0441: +++o
0442: +++o
0443: +++o
0444: +++o
0445: +++o
0446: +++o
0447: +++o
0448: +++o
0449: +++o
0450: +++o
0451: +++o
0452: +++o
0453: +++o
0454: +++o
0455: +++o
0456: +++o
0457: +++o
0458: +++o
0459: +++o
0460: +++o
0461: +++o
0462: +++o
0463: +++o
0464: +++o
0465: +++o

0466: +++o
0467: +++o
0468: +++o
0469: +++o
0470: +++o
0471: +++o
0472: +++o
0473: +++o
0474: +++o
0475: +++o
0476: +++o
0477: +++o
0478: +++o
0479: +++o
0480: +++o
0481: +++o
0482: +++o
0483: +++o
0484: +++o
0485: +++o
0486: +++o
0487: +++o
0488: +++o
0489: +++o
0490: +++o
0491: +++o
0492: +++o
0493: +++o
0494: +++o
0495: +++o
0496: +++o
0497: +++o
0498: +++o
0499: +++o
0500: +++o
0501: +++o
0502: +++o
0503: +++o
0504: +++o
0505: +++o
0506: +++o
0507: +++o
0508: +++o
0509: +++o
0510: +++o
0511: +++o
0512: +++o
0513: +++o
0514: +++o
0515: +++o
0516: +++o
0517: +++o
0518: +++o
0519: +++o
0520: +++o
0521: +++o
0522: +++o
0523: +++o
0524: +++o
0525: +++o
0526: +++o
0527: +++o
0528: +++o
0529: +++o
0530: +++o
0531: +++o
0532: +++o
0533: +++o
0534: +++o
0535: +++o
0536: +++o
0537: +++o
0538: +++o
0539: +++o
0540: +++o
0541: +++o
0542: +++o
0543: +++o
0544: +++o
0545: +++o
0546: +++o
0547: +++o
0548: +++o
0549: +++o
0550: +++o
0551: +++o
0552: +++o
0553: +++o
0554: +++o
0555: +++o
0556: +++o
0557: +++o
0558: +++o
0559: +++o
0560: +++o
0561: +++o
0562: +++o
0563: +++o
0564: +++o
0565: +++o
0566: +++o
0567: +++o
0568: +++o
0569: +++o
0570: +++o
0571: +++o
0572: +++o
0573: +++o
0574: +++o
0575: +++o
0576: +++o
0577: +++o
0578: +++o
0579: +++o
0580: +++o
0581: +++o
0582: +++o
0583: +++o
0584: +++o
0585: +++o
0586: +++o
0587: +++o
0588: +++o
0589: +++o
0590: +++o
0591: +++o
0592: +++o
0593: +++o
0594: +++o
0595: +++o
0596: +++o
0597: +++o
0598: +++o
0599: +++o
0600: +++o
0601: +++o
0602: +++o
0603: +++o
0604: +++o
0605: +++o
0606: +++o
0607: +++o
0608: +++o
0609: +++o
0610: +++o
0611: +++o

0612: +++o
0613: +++o
0614: +++o
0615: +++o
0616: +++o
0617: +++o
0618: +++o
0619: +++o
0620: +++o
0621: +++o
0622: +++o
0623: +++o
0624: +++o
0625: +++o
0626: +++o
0627: +++o
0628: +++o
0629: +++o
0630: +++o
0631: +++o
0632: +++o
0633: +++o
0634: +++o
0635: +++o
0636: +++o
0637: +++o
0638: +++o
0639: +++o
0640: +++o
0641: +++o
0642: +++o
0643: +++o
0644: +++o
0645: +++o
0646: +++o
0647: +++o
0648: +++o
0649: +++o
0650: +++o
0651: +++o
0652: +++o
0653: +++o
0654: +++o
0655: +++o
0656: +++o
0657: +++o
0658: +++o
0659: +++o
0660: +++o
0661: +++o
0662: +++o
0663: +++o
0664: +++o
0665: +++o
0666: +++o
0667: +++o
0668: +++o
0669: +++o
0670: +++o
0671: +++o
0672: +++o
0673: +++o
0674: +++o
0675: +++o
0676: +++o
0677: +++o
0678: +++o
0679: +++o
0680: +++o
0681: +++o
0682: +++o
0683: +++o
0684: +++o
0685: +++o
0686: +++o
0687: +++o
0688: +++o
0689: +++o
0690: +++o
0691: +++o
0692: +++o
0693: +++o
0694: +++o
0695: +++o
0696: +++o
0697: +++o
0698: +++o
0699: +++o
0700: +++o
0701: +++o
0702: +++o
0703: +++o
0704: +++o
0705: +++o
0706: +++o
0707: +++o
0708: +++o
0709: +++o
0710: +++o
0711: +++o
0712: +++o
0713: +++o
0714: +++o
0715: +++o
0716: +++o
0717: +++o
0718: +++o
0719: +++o
0720: +++o
0721: +++o
0722: +++o
0723: +++o
0724: +++o
0725: +++o
0726: +++o
0727: +++o
0728: +++o
0729: +++o
0730: +++o
0731: +++o
0732: +++o
0733: +++o
0734: +++o
0735: +++o
0736: +++o
0737: +++o
0738: +++o
0739: +++o
0740: +++o
0741: +++o
0742: +++o
0743: +++o
0744: +++o
0745: +++o
0746: +++o
0747: +++o
0748: +++o
0749: +++o
0750: +++o
0751: +++o
0752: +++o
0753: +++o
0754: +++o
0755: +++o
0756: +++o
0757: +++o

0758: +++o
0759: +++o
0760: +++o
0761: +++o
0762: +++o
0763: +++o
0764: +++o
0765: +++o
0766: +++o
0767: +++o
0768: +++o
0769: +++o
0770: +++o
0771: +++o
0772: +++o
0773: +++o
0774: +++o
0775: +++o
0776: +++o
0777: +++o
0778: +++o
0779: +++o
0780: +++o
0781: +++o
0782: +++o
0783: +++o
0784: +++o
0785: +++o
0786: +++o
0787: +++o
0788: +++o
0789: +++o
0790: +++o
0791: +++o
0792: +++o
0793: +++o
0794: +++o
0795: +++o
0796: +++o
0797: +++o
0798: +++o
0799: +++o
0800: +++o
0801: +++o
0802: +++o
0803: +++o
0804: +++o
0805: +++o
0806: +++o
0807: +++o
0808: +++o
0809: +++o
0810: +++o
0811: +++o
0812: +++o
0813: +++o
0814: +++o
0815: +++o
0816: +++o
0817: +++o
0818: +++o
0819: +++o
0820: +++o
0821: +++o
0822: +++o
0823: +++o
0824: +++o
0825: +++o
0826: +++o
0827: +++o
0828: +++o
0829: +++o
0830: +++o
0831: +++o
0832: +++o
0833: +++o
0834: +++o
0835: +++o
0836: +++o
0837: +++o
0838: +++o
0839: +++o
0840: +++o
0841: +++o
0842: +++o
0843: +++o
0844: +++o
0845: +++o
0846: +++o
0847: +++o
0848: +++o
0849: +++o
0850: +++o
0851: +++o
0852: +++o
0853: +++o
0854: +++o
0855: +++o
0856: +++o
0857: +++o
0858: +++o
0859: +++o
0860: +++o
0861: +++o
0862: +++o
0863: +++o
0864: +++o
0865: +++o
0866: +++o
0867: +++o
0868: +++o
0869: +++o
0870: +++o
0871: +++o
0872: +++o
0873: +++o
0874: +++o
0875: +++o
0876: +++o
0877: +++o
0878: +++o
0879: +++o
0880: +++o
0881: +++o
0882: +++o
0883: +++o
0884: +++o
0885: +++o
0886: +++o
0887: +++o
0888: +++o
0889: +++o
0890: +++o
0891: +++o
0892: +++o
0893: +++o
0894: +++o
0895: +++o
0896: +++o
0897: +++o
0898: +++o
0899: +++o
0900: +++o
0901: +++o
0902: +++o
0903: +++o

```
0904: +++o
0905: +++o
0906: +++o
0907: +++o
0908: +++o
0909: +++o
0910: +++o
0911: +++o
0912: +++o
0913: +++o
0914: +++o
0915: +++o
0916: +++o
0917: +++o
0918: +++o
0919: +++o
0920: +++o
0921: +++o
0922: +++o
0923: +++o
0924: +++o
0925: +++o
0926: +++o
0927: +++o
0928: +++o
0929: +++o
0930: +++o
0931: +++o
0932: +++o
0933: +++o
0934: +++o
0935: +++o
0936: +++o
0937: +++o
0938: +++o
0939: +++o
0940: +++o
0941: +++o
0942: +++o
0943: +++o
0944: +++o
0945: +++o
0946: +++o
0947: +++o
0948: +++o
0949: +++o
0950: +++o
0951: +++o
0952: +++o
0953: +++o
0954: +++o
0955: +++o
0956: +++o
0957: +++o
0958: +++o
0959: +++o
0960: +++o
0961: +++o
0962: +++o
0963: +++o
0964: +++o
0965: +++o
0966: +++o
0967: +++o
0968: +++o
0969: +++o
0970: +++o
0971: +++o
0972: +++o
0973: +++o
0974: +++o
0975: +++o
0976: +++o
0977: +++o
0978: +++o
0979: +++o
0980: +++o
0981: +++o
0982: +++o
0983: +++o
0984: +++o
0985: +++o
0986: +++o
0987: +++o
0988: +++o
0989: +++o
0990: +++o
0991: +++o
0992: +++o
0993: +++o
0994: +++o
0995: +++o
0996: +++o
0997: +++o
0998: +++o
0999: +++o
1000: +++o
1001: +++o
1002: +++o
1003: +++o
1004: +++o
1005: +++o
1006: +++o
1007: +++o
1008: +++o
1009: +++o
1010: +++o
1011: +++o
1012: +++o
1013: +++o
1014: +++o
1015: +++o
1016: +++o
1017: +++o
1018: +++o
1019: +++o
1020: +++o
1021: +++o
1022: +++o
1023: +++o
1024: +++o
octave:19> diary off
```

RESULTS OF TEST3B.M

```
octave:4> date
ans = [redacted]
octave:5> clock
ans =

[redacted]

octave:6> type test3b
test3b is the script file: /1093-2/test3b.m

## TEST3B.M
## Block size is 10 bits. Input is taken from set Z:{1,2,...1024}
## Because of EAS-invented "pseudogroup" operation, output also
## falls in set Z.
## Keys are also taken from set Z - any set element is OK.

## This test analyzes outputs for a given input over all
## possible keys.

[redacted]
## Written for Octave (GNU MATLAB alternative)
[redacted]

## No paging - want current screen output
page_screen_output=0;

## Set values defining set and underlying group order
N = 10; M = 2^N; # M = 1024
k = 7; p = M+k; # p = 1031 (prime)

## Define vector with elements of set Z
v = linspace(1,M,M);

## Define vector of skip/repeat counts
cc = zeros(1,M);

disp(['Test for each input value in set 1,2,...',num2str(M)]);
disp('--------------------------------------------------------');

## For all possible input values in Z...
for i = 1:M

  ## Show progress
  disp(['Encrypting with input value y=',num2str(i),'...']);

  ## Set input value for this iteration
  x = v(i);

  ## Encrypt input value with all keys in set Z
  for j = 1:M
    b(j) = encrypt(x,v(j),N,k);
  endfor
```

```
  disp(['Output set: min=',num2str(min(b)),', max=',num2str(max(b))]);
  disp('');

  ## Identify any skipped or repeated set elements
  ## with vector of index numbers
  b1 = sort(b); # Sort ascending
  b2 = [diff(b1)']; # Should be all 1's...
  b2 = b2~=1; # ...so 1's indicate skips/repeats

  Nsr = sum(b2); # Count of skips/repeats

  b3 = b2 .* v(1:M-1); # map index numbers to skips/repeats
  b3 = sort(b3); # Sort ascending
  b4 = b3(M-Nsr:M-1); # Select only skips/repeats

  if (Nsr > 0)

    disp(['There are ',num2str(Nsr),' skips & repeats, at:']);
    disp(b4);
    disp('-------------------------------------------------------');

    c = zeros(6,Nsr); # Start with empty ("0") matrix
    for j = 1:Nsr
      k1 = max([1 b4(j)-2]);
      k2 = min([b4(j)+3 M]);
      c(1:k2-k1+1,j) = b1(k1:k2);
    endfor

    disp(c)

  endif

  cc(i) = Nsr; # Add this count to vector
  disp(['Maximum skips & repeats for a given input (so far):',num2str(max(cc))]);
  disp('');

endfor

. . .

octave:9> test3b
Test for each input value in set 1,2,...1024
-------------------------------------------------------
Encrypting with input value y=1...
Output set: min=1, max=1024

Maximum skips & repeats for a given input (so far):0

Encrypting with input value y=2...
Output set: min=1, max=1024

There are 6 skips & repeats, at:
    10   518   520  1021  1022  1023
-------------------------------------------------------
     8   515   517  1016  1017  1018
     9   516   517  1017  1018  1020
    10   517   518  1018  1020  1022
    10   517   518  1020  1022  1024
```

```
    11   518   519  1022  1024     0
    12   518   520  1024     0     0
Maximum skips & repeats for a given input (so far):6

Encrypting with input value y=3...
Output set: min=1, max=1024

There are 8 skips & repeats, at:
    10   346   690   692  1016  1018  1020  1022
------------------------------------------------------------
     8   343   686   688  1010  1012  1015  1018
     9   344   687   688  1011  1014  1017  1020
    10   345   688   689  1012  1015  1018  1021
    10   345   688   689  1014  1017  1020  1023
    11   346   689   690  1015  1018  1021  1024
    12   347   689   691  1017  1020  1023     0
Maximum skips & repeats for a given input (so far):8

Encrypting with input value y=4...
Output set: min=1, max=1024

There are 10 skips & repeats, at:
     3   260   519   523   778  1011  1014  1017  1020  1023
------------------------------------------------------------
     1   257   515   518   772  1004  1008  1012  1016  1020
     2   258   516   519   773  1005  1009  1013  1017  1021
     3   259   517   520   774  1006  1010  1014  1018  1022
     3   259   517   520   774  1008  1012  1016  1020  1024
     4   260   518   521   775  1009  1013  1017  1021     0
     5   261   519   522   776  1010  1014  1018  1022     0
Maximum skips & repeats for a given input (so far):10

Encrypting with input value y=5...
Output set: min=1, max=1024

There are 10 skips & repeats, at:
     6   212   418   624   830  1005  1009  1013  1017  1021
------------------------------------------------------------
     4   209   414   619   824   998  1003  1008  1013  1018
     5   210   415   620   825   999  1004  1009  1014  1019
     6   211   416   621   826  1000  1005  1010  1015  1020
     6   211   416   621   826  1002  1007  1012  1017  1022
     7   212   417   622   827  1003  1008  1013  1018  1023
     8   213   418   623   828  1004  1009  1014  1019  1024
Maximum skips & repeats for a given input (so far):10

Encrypting with input value y=6...
Output set: min=1, max=1024

There are 10 skips & repeats, at:
   176   346   518   691   864   999  1004  1009  1014  1019
------------------------------------------------------------
   174   343   514   686   858   992   998  1004  1010  1016
   175   344   515   687   859   993   999  1005  1011  1017
   176   345   516   688   860   994  1000  1006  1012  1018
   176   345   516   688   860   996  1002  1008  1014  1020
   177   346   517   689   861   997  1003  1009  1015  1021
   178   347   518   690   862   998  1004  1010  1016  1022
```

Maximum skips & repeats for a given input (so far):10

Encrypting with input value y=7...
Output set: min=1, max=1023

There are 11 skips & repeats, at:

149	297	445	593	741	889	994	1000	1006	1012	1018
147	294	441	588	735	882	986	993	1000	1007	1014
148	295	442	589	736	883	987	994	1001	1008	1015
149	296	443	590	737	884	988	995	1002	1009	1016
149	296	443	590	737	884	990	997	1004	1011	1018
150	297	444	591	738	885	991	998	1005	1012	1019
151	298	445	592	739	886	992	999	1006	1013	1020

Maximum skips & repeats for a given input (so far):11

Encrypting with input value y=8...
Output set: min=1, max=1024

There are 12 skips & repeats, at:

3	4	261	520	521	779	988	995	1002	1009	1016	1023
1	2	257	515	516	772	980	988	996	1004	1012	1020
2	3	258	516	517	773	981	989	997	1005	1013	1021
3	3	259	517	517	774	982	990	998	1006	1014	1022
3	3	259	517	517	774	984	992	1000	1008	1016	1024
3	4	260	517	518	775	985	993	1001	1009	1017	0
4	5	261	518	519	776	986	994	1002	1010	1018	0

Maximum skips & repeats for a given input (so far):12

Encrypting with input value y=9...
Output set: min=1, max=1024

There are 12 skips & repeats, at:

2	346	347	464	692	922	982	990	998	1006	1014	1022
1	343	344	459	686	915	974	983	992	1001	1010	1019
2	344	345	460	687	916	975	984	993	1002	1011	1020
2	345	345	461	688	917	976	985	994	1003	1012	1021
3	345	345	461	688	917	978	987	996	1005	1014	1023
4	345	346	462	689	918	979	988	997	1006	1015	1024
0	346	347	463	690	919	980	989	998	1007	1016	0

Maximum skips & repeats for a given input (so far):12

Encrypting with input value y=10...
Output set: min=1, max=1024

There are 12 skips & repeats, at:

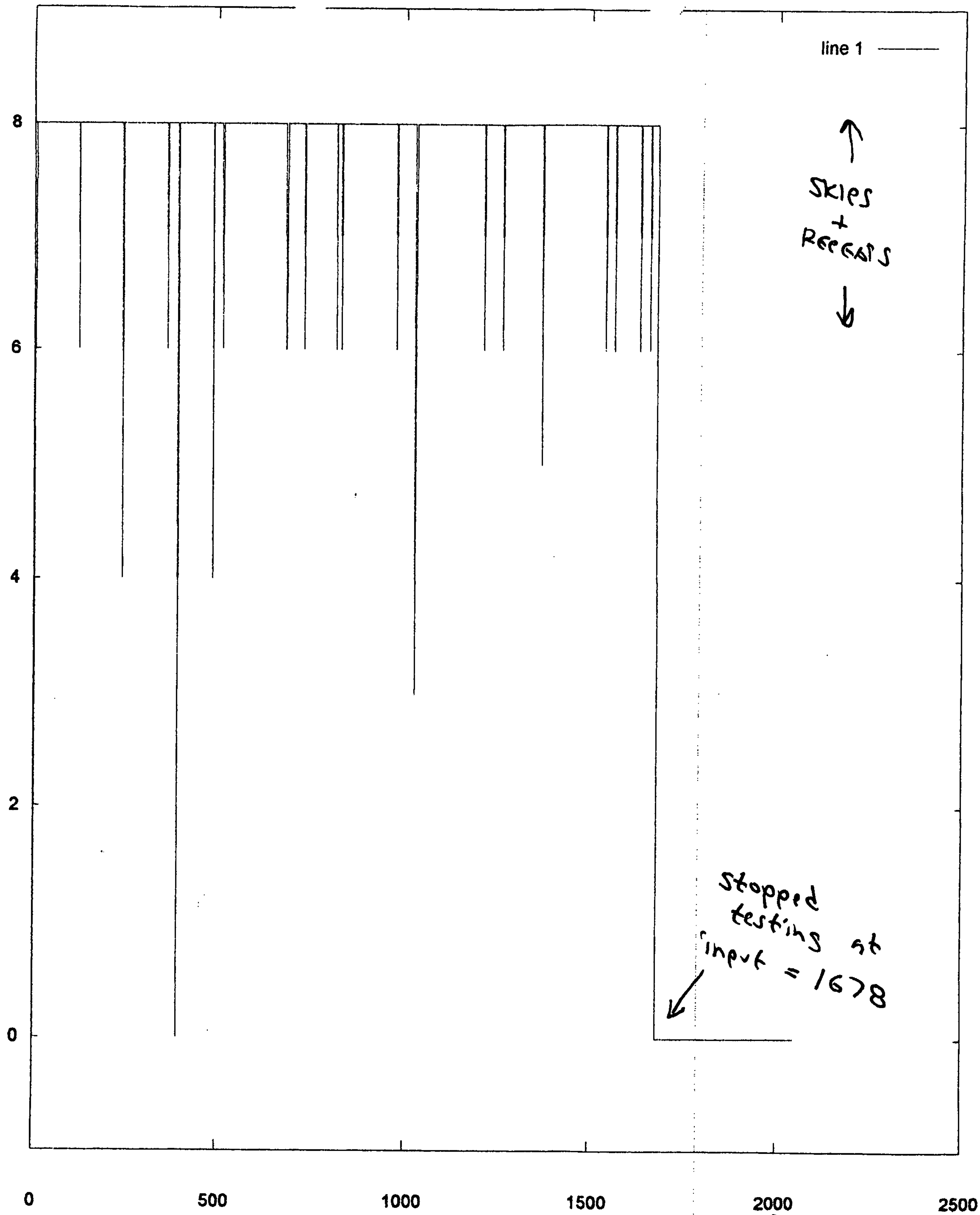
2	4	416	520	829	933	976	985	994	1003	1012	1021
1	2	412	515	823	926	968	978	988	998	1008	1018
2	2	413	516	824	927	969	979	989	999	1009	1019
2	3	414	517	825	928	970	980	990	1000	1010	1020
3	3	414	517	825	928	972	982	992	1002	1012	1022
3	4	415	518	826	929	973	983	993	1003	1013	1023
0	5	416	519	827	930	974	984	994	1004	1014	1024

Maximum skips & repeats for a given input (so far):12

```
     17    35    53    71    89   107   114   342   458   687   687   859
     19    37    55    73    91   109   114   342   458   687   687   859
     20    38    56    74    92   110   115   343   459   687   688   860
     21    39    57    75    93   111   116   344   460   688   689   861
Maximum skips & repeats for a given input (so far):12

Encrypting with input value y=1014...
Output set: min=1, max=1024

There are 12 skips & repeats, at:
    16    32    48    64    80    96   236   479   661   845   907   969
--------------------------------------------------------------------------
     14    31    48    65    82    99   240   482   663   846   907   968
     15    32    49    66    83   100   241   483   664   847   908   969
     16    33    50    67    84   101   242   484   665   848   909   970
     18    35    52    69    86   103   242   484   665   848   909   970
     19    36    53    70    87   104   243   485   666   849   910   971
     20    37    54    71    88   105   244   486   667   850   911   972
Maximum skips & repeats for a given input (so far):12

Encrypting with input value y=1015...
Output set: min=1, max=1024

There are 12 skips & repeats, at:
    15    30    45    60    75    90   121   188   511   641   771   901
--------------------------------------------------------------------------
     13    29    45    61    77    93   125   191   513   642   771   900
     14    30    46    62    78    94   126   192   514   643   772   901
     15    31    47    63    79    95   127   193   515   644   773   902
     17    33    49    65    81    97   127   193   515   644   773   902
     18    34    50    66    82    98   128   194   516   645   774   903
     19    35    51    67    83    99   129   195   517   646   775   904
Maximum skips & repeats for a given input (so far):12

Encrypting with input value y=1016...
Output set: min=1, max=1024

There are 12 skips & repeats, at:
    14    28    42    56    70    84   199   338   614   752   891   961
--------------------------------------------------------------------------
     12    27    42    57    72    87   203   341   616   753   891   960
     13    28    43    58    73    88   204   342   617   754   892   961
     14    29    44    59    74    89   205   343   618   755   893   962
     16    31    46    61    76    91   205   343   618   755   893   962
     17    32    47    62    77    92   206   344   619   756   894   963
     18    33    48    63    78    93   207   345   620   757   895   964
Maximum skips & repeats for a given input (so far):12

Encrypting with input value y=1017...
Output set: min=1, max=1024

There are 12 skips & repeats, at:
    13    26    39    52    65    78   140   436   437   658   734   956
--------------------------------------------------------------------------
     11    25    39    53    67    81   144   439   440   659   734   955
     12    26    40    54    68    82   145   440   441   660   735   956
     13    27    41    55    69    83   146   441   441   661   736   957
     15    29    43    57    71    85   146   441   441   661   736   957
```

```
   16   30   44   58   72   86  147  441  442  662  737  958
   17   31   45   59   73   87  148  442  443  663  738  959
Maximum skips & repeats for a given input (so far):12

Encrypting with input value y=1018...
Output set: min=1, max=1024

There are 12 skips & repeats, at:
   12   24   36   48   60   72   73  231  233  631  632  871
-------------------------------------------------------------
   10   23   36   49   62   75   76  234  236  632  633  870
   11   24   37   50   63   76   77  235  236  633  634  871
   12   25   38   51   64   77   77  236  237  634  634  872
   14   27   40   53   66   77   79  236  237  634  634  872
   15   28   41   54   67   79   80  237  238  634  635  873
   16   29   42   55   68   80   81  237  239  635  636  874
Maximum skips & repeats for a given input (so far):12

Encrypting with input value y=1019...
Output set: min=1, max=1024

There are 12 skips & repeats, at:
   11   22   33   44   55   66  336  508  682  769  857  944
-------------------------------------------------------------
    9   21   33   45   57   69  340  511  684  770  857  943
   10   22   34   46   58   70  341  512  685  771  858  944
   11   23   35   47   59   71  342  513  686  772  859  945
   13   25   37   49   61   73  342  513  686  772  859  945
   14   26   38   50   62   74  343  514  687  773  860  946
   15   27   39   51   63   75  344  515  688  774  861  947
Maximum skips & repeats for a given input (so far):12

Encrypting with input value y=1020...
Output set: min=1, max=1024

There are 12 skips & repeats, at:
   10   20   30   40   50   60   85  182  275  652  747  936
-------------------------------------------------------------
    8   19   30   41   52   63   89  185  277  653  747  935
    9   20   31   42   53   64   90  186  278  654  748  936
   10   21   32   43   54   65   91  187  279  655  749  937
   12   23   34   45   56   67   91  187  279  655  749  937
   13   24   35   46   57   68   92  188  280  656  750  938
   14   25   36   47   58   69   93  189  281  657  751  939
Maximum skips & repeats for a given input (so far):12

Encrypting with input value y=1021...
Output set: min=1, max=1024

There are 12 skips & repeats, at:
    9   18   27   36   45   54  199  407  510  615  616  823
-------------------------------------------------------------
    7   17   27   37   47   57  203  410  512  616  617  822
    8   18   28   38   48   58  204  411  513  617  618  823
    9   19   29   39   49   59  205  412  514  618  618  824
   11   21   31   41   51   61  205  412  514  618  618  824
   12   22   32   42   52   62  206  413  515  618  619  825
   13   23   33   43   53   63  207  414  516  619  620  826
```

```
Maximum skips & repeats for a given input (so far):12

Encrypting with input value y=1022...
Output set: min=1, max=1024

There are 12 skips & repeats, at:
     8   16   24   32   40   48  222  336  567  684  799  915
-----------------------------------------------------------
     6   15   24   33   42   51  226  339  569  685  799  914
     7   16   25   34   43   52  227  340  570  686  800  915
     8   17   26   35   44   53  228  341  571  687  801  916
    10   19   28   37   46   55  228  341  571  687  801  916
    11   20   29   38   47   56  229  342  572  688  802  917
    12   21   30   39   48   57  230  343  573  689  803  918
Maximum skips & repeats for a given input (so far):12

Encrypting with input value y=1023...
Output set: min=1, max=1024

There are 12 skips & repeats, at:
     7   14   21   28   35   42  248  249  511  641  771  901
-----------------------------------------------------------
     5   13   21   29   37   45  252  253  513  642  771  900
     6   14   22   30   38   46  253  254  514  643  772  901
     7   15   23   31   39   47  254  254  515  644  773  902
     9   17   25   33   41   49  254  254  515  644  773  902
    10   18   26   34   42   50  254  255  516  645  774  903
    11   19   27   35   43   51  255  256  517  646  775  904
Maximum skips & repeats for a given input (so far):12

Encrypting with input value y=1024...
Output set: min=1, max=1024

There are 12 skips & repeats, at:
     6   12   18   24   30   36  137  286  435  584  733  882
-----------------------------------------------------------
     4   11   18   25   32   39  141  289  437  585  733  881
     5   12   19   26   33   40  142  290  438  586  734  882
     6   13   20   27   34   41  143  291  439  587  735  883
     8   15   22   29   36   43  143  291  439  587  735  883
     9   16   23   30   37   44  144  292  440  588  736  884
    10   17   24   31   38   45  145  293  441  589  737  885
Maximum skips & repeats for a given input (so far):12

octave:10> date
ans = [redacted]
octave:11> clock
ans =

[redacted]

octave:12> diary off
```



← lnew →

TEST4B.M
WITH M = 2048
p = 2053

W-1

Pronounceable Passphrase Worksheet

by Edwin A. Suominen

Digit Content	Min. Digits	Entropy	Example Passphrase
Alternating consonants (C), Vowels (V) C,13: {b,d,g,h,k,l,m,n,p,r,s,t,z} V,5: {a,e,i,o,u}	C: 11 V: 10	About 64 bits, for minimum digits	nihudezo dogiz pozubume

The vowels and phonetically distinct consonanants below are pseudorandomly distributed, created using alternating pseudorandom lookups to a list of consonants b,d,g,h,k,l,m,n,p,r,s,t,z and vowels a,e,i,o,u. Alternating randomly selected consonants and randomly selected vowels from the array below form passphrases that have a faintly Oriental or African sound to them, and are more memorable than random alphanumerics. Consonants c,f,j,q,v,w,x,y are omitted because the passphrases tend to have a more distinct sound and are easier to pronounce without them.

You should split the consonant/vowel pairs into groups to make the passphrase pronounceable and thus more memorable. The suggested way of grouping the minimum 11 consonants and 10 vowels is as follows: CVCVCVCV CVCVC CVCVCVCV. Note that the middle group begins and ends with a consonant. The resulting passphrase has a distinct sound that makes you wonder if the "words" show up in some foreign language even though they're just groups of randomly chosen letters.

Unbend a paper clip slightly, repeatedly toss the clip onto a printout of this page without aiming it anywhere in particular, and select the consonant/vowel pair to which the unbent end comes closest to get the next digit in your consonant-vowel sequence. Don't use both digits from a pair – each digit in your passphrase needs its own toss. With good random tosses, you can expect the clip to bounce outside the array of digits below about half the time. Just toss again. Don't aim at any particular region.

```
ne ze nu hi ni ba hu pi pu lu me ki lu ge le mu nu ko se ze ta ba ga ro ta be ko te
lo ru hu gi ra ga ro do ro zu tu si mu pu lo bi ze tu ha ti ru to ni ze bu be si ma
ro le hi ni bo ma do hu ma lu bu zi ru sa ta zo po go ta hi ri bi te la me go ho si
go ka hi bo li ro mi ta mo ki ku hu ri le da ra za mi le mi da ra zu ki ke bi mi do
mo ta tu ta ra si la go ki su ki mi re ba le so to za ba te hi ri da go za ka sa za
mo na la me ku bu mu me no ke ri be ho tu mu ki no re ho pe mi hu ho pi ge ro to se
go ka ze ri du sa de bi si de mi pi se po le mi bu re pe du so lu la ri mo mo bo go
to ge ne go lo ru sa da ma ga ni pe si se bu di pi da pu ta la ku hu ko go za si de
tu me po ro to pe ku hu ra ha bu zo ti na ba ma se te pi do to ke so do re ru ze ru
to lo ne ki ta ha go go li ra po de na du ba su po ka lu bu hi be za la ke bo ne mi
he ni te ma ni pu ho ru ne da nu de ge ge so ge lu re ho bi po li ma ri su pa te du
zi ta ma lu ti bu ha ba hu mu su za ko to ga te tu ru la ki ru ze po ni pi ho sa pa
pa bu pi za du go ga su mu re su ni re hi bi ko po la zu ri ku ka zi ba pa tu di da
ma mo di hu po ro ho ku se pa bo la ga ne me pi pu gu he li ha gi bo he hu so ru pa
se be pe ga se ni lo bo se ka be gi de zi di me mu lu pe ku tu go pe hi di bi de ne
mo mi ki ni ka ha nu ka za do he ni da du ri se du lo se tu pe to ti hu na ru se ta
na ho bu tu ze ge ru ru ba ga li ro so mu du he da go mu da ne tu mu pi ne bu ka la
li le he me gi ru ge hi lo po ho pe no mu pe ta lo re na di gi ko ze re pi ga ba ki
ke su zo ra no di lu du be za na po do su ra do si he me bi ga ra ha do za ru ku ri
zi bu do lo ga ki ha zi ka ru re ke ti lo zu zu gu bu ma bo hi he si du ne hu da ka
te ne na zi hu si da no mo te bu ko tu ro mu ne ma su li ba ga te se mo bi re si tu
ta pe bo bu ma tu ge be ri no pa do ri za ra ho so to da ze ze lo he mi ha la de me
pu ha no te de la zi hu gi ro te pa mo bu mu pe da sa pe na pi ti ru ro pi go li ku
gi gi pu za pe go ti lo pu du su mo ta ki ha ge ka ke pa tu po hi be lu sa bo li so
ko ra ri zi po ra la ze pe ru zu ra mu zu du ka lo ku zo po mo no he ze su ho le gi
te go pu ka ga la ki lu pa ti na ga gi ba ka pi ka su gi mo no si te li gi se zi ze
ze za lu ge ro zi ra do bo ma zo mi ke ze la bu ru be ki te ga ni go go gi ga no ha
si bu be mi ta pa de ke ta lu ru ma ra si se te me pe pu gi mu me li bi bi hi zu me
de he ku zu nu gi pi zi ga ka ze lo re ru ha zu ta ku pi po me la da mu li de he be
pi le mu ko ki nu ke ga hu la la ri po su ri na ru no ta ro ho zi so ru ri go su hu
gi ra ru pu mu gi le ga ti ne go le ba da gi si si ni ra zu ma ha bu le se ze ro go
pi no bu ne nu ma si re ka gu zo si bo sa be zo si ko po li gi ra pe ne gu ri la ki
go ka ze ne ro mo bi nu he no hu ne ha lo ne le hu hi so du de li ta ze li hu de do
ga ha bi he su ku pe de he hu ta so zo he bu ti hi gi ta ti ze pe gu ni ka ra su pi
lu ge pe ka pu na da mu ti du mo pi de pu pa ri si bi sa mi li ko mi be ge ku su po
me mi tu ze ge ki gu hu hi lo tu da mu to ru gi ru ra mo na ma ba si se ri ro ro ka
po ge do sa pu ma so na ba la li ka ne ba di te zo so ru mi se ni gu na ta di zi ri
ga ga ri no ho sa ne mo le hi he re ta ne go ki de pi de gi mu du tu bo to ki po di
ma ge bo se ne hi ni ma go li le ra ri mu si lo gi zo ku bo ro ze po to do go re tu
ga ka la ki za ti me te zi ho mi ta no ne lo za ma mo te si sa ba da ku ko be ki go
zi zu tu pe ma ba zu lo ra ge hu pa te ra ra ko zu ni se he to ma se gu gu ba bi te
pe zo ne no gu ne bi ga ra na ra na la la do nu du za ro li li mu ne ru ni he te ma
```

Description	Digit Space #1	No. of #1 Digits	Entropy #1 (bits)	Digit Space #2	No. of #2 Digits	Entropy #2 (bits)	Digit Space #3	No. of #3 Digits	Entropy #3 (bits)	Total Entropy
6 word DICEWARE (http://diceware.com)	7776	6	77.54888	1	0	0	1	0	0	**77.5**
14 alphanumeric digits except lowercase "L" (can group 5-4-5 or 4-6-4)	35	14	71.80996	1	0	0	1	0	0	**71.8**
13 alphanumeric digits except lowercase "L" (can group 4-5-4)	35	13	66.68068	1	0	0	1	0	0	**66.7**
Pairs of phonetically distinct consonants "b,d,g,h,k,l,m,n,p,r,s,t,z") followed by vowel, arranged as follows: cvcvcvcv cvcvc cvcvcvcv	13	11	40.70484	5	10	23.21928	1	0	0	**63.9**
12 alphanumeric digits except lowercase "L" separated randomly into three groups	35	12	61.5514	11	1	3.459432	1	0	0	**65.0**
5 word DICEWARE	7776	5	64.62406	1	0	0	1	0	0	**64.6**
12 alphanumeric digits except lowercase "L" (can be grouped 4-4-4)	35	12	61.5514	1	0	0	1	0	0	**61.6**
4 words from 70K dictionary	70000	4	64.38027	1	0	0	1	0	0	**64.4**
Three dates, Month, Day, Year	12	3	10.75489	30	3	14.72067	500	3	26.89735	**52.4**
Random first name (10K) plus M.I. Plus last name (50K)	10000	1	13.28771	50000	1	15.60964	26	1	4.70044	**33.6**
Random street address	20000	1	14.28771	8	1	3	10000	1	13.28771	30.6
++	No.			NSEW		30.6	20	1	4.321928	**34.9**
(e.g.,	1	0	0	1	0	0	1	0	0	**0.0**
Passwords for smartcards)	1	0	0	1	0	0	1	0	0	**0.0**
Pairs of phonetically distinct consonants "b,d,g,h,k,l,m,n,p,r,s,t,z") followed by vowel, arranged as follows: cvcvcvcv cvcvc cvcvcvcv	13	4	14.80176	5	4	9.287712	1	0	0	**24.1**
5 alphanumeric digits except lowercase "L"	35	5	25.64642	1	0	0	1	0	0	**25.6**
(e.g.,	1	0	0	1	0	0	1	0	0	**0.0**
Medium Security for use with secure delay)	1	0	0	1	0	0	1	0	0	**0.0**
Pairs of phonetically distinct consonants "b,d,g,h,k,l,m,n,p,r,s,t,z") followed by vowel, arranged as follows: cvcvcvcv cvcvc cvcvcvcv	13	8	29.60352	5	8	18.57542	1	0	0	**48.2**
Pairs of phonetically distinct consonants "b,d,g,h,k,n,p,s,t,z") followed by vowel, arranged as follows: cvcvcvcv cvcvc cvcvcvcv	10	8	26.57542	5	8	18.57542	1	0	0	**45.2**
Groups of numbers 1000-9192 = {0-8192}+1000	8192	4	52	1	0	0	1	0	0	52
	1	0	0	1	0	0	1	0	0	**0.0**
	1	0	0	1	0	0	1	0	0	**0.0**

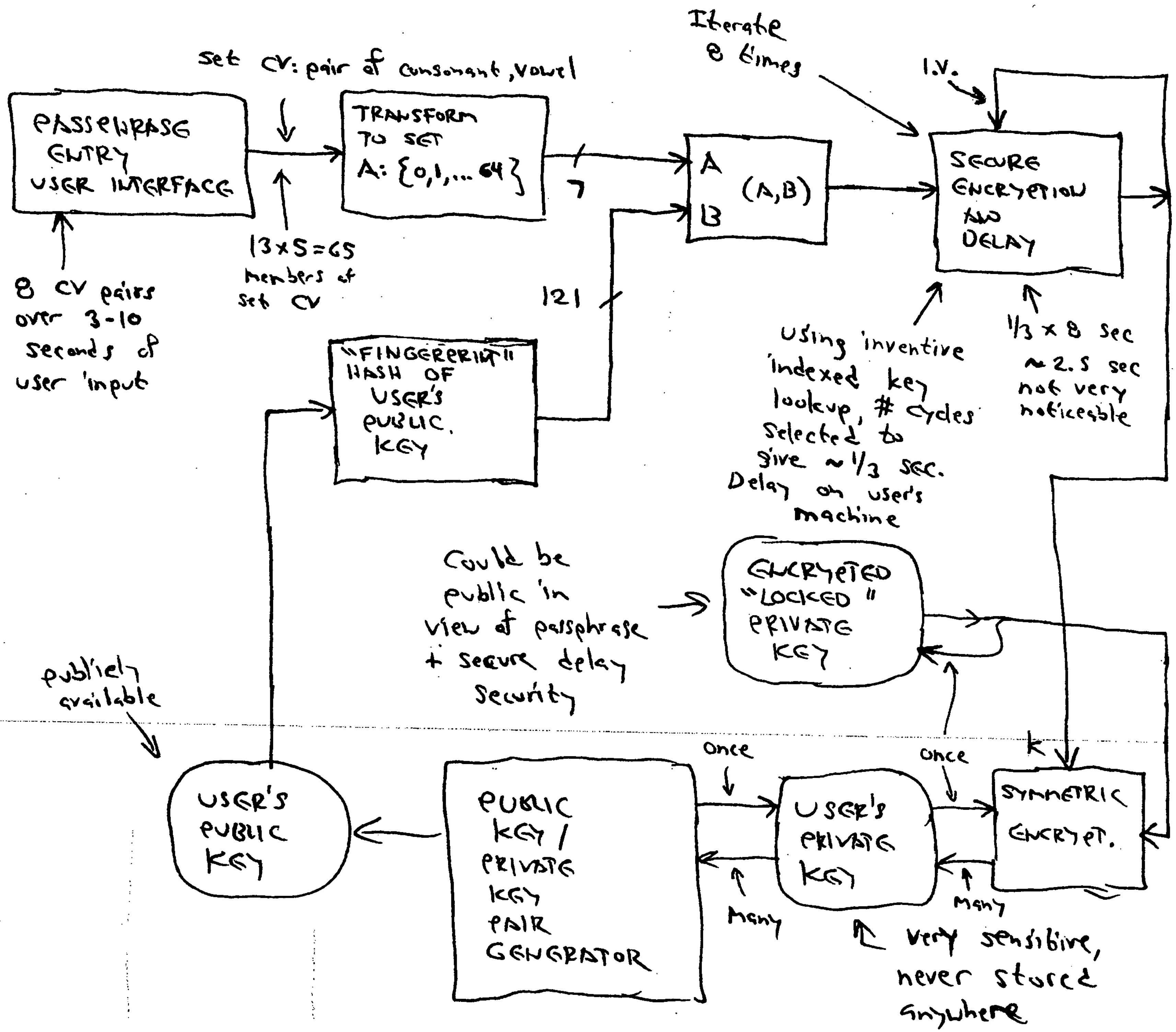
SECURE PASSPHRASE

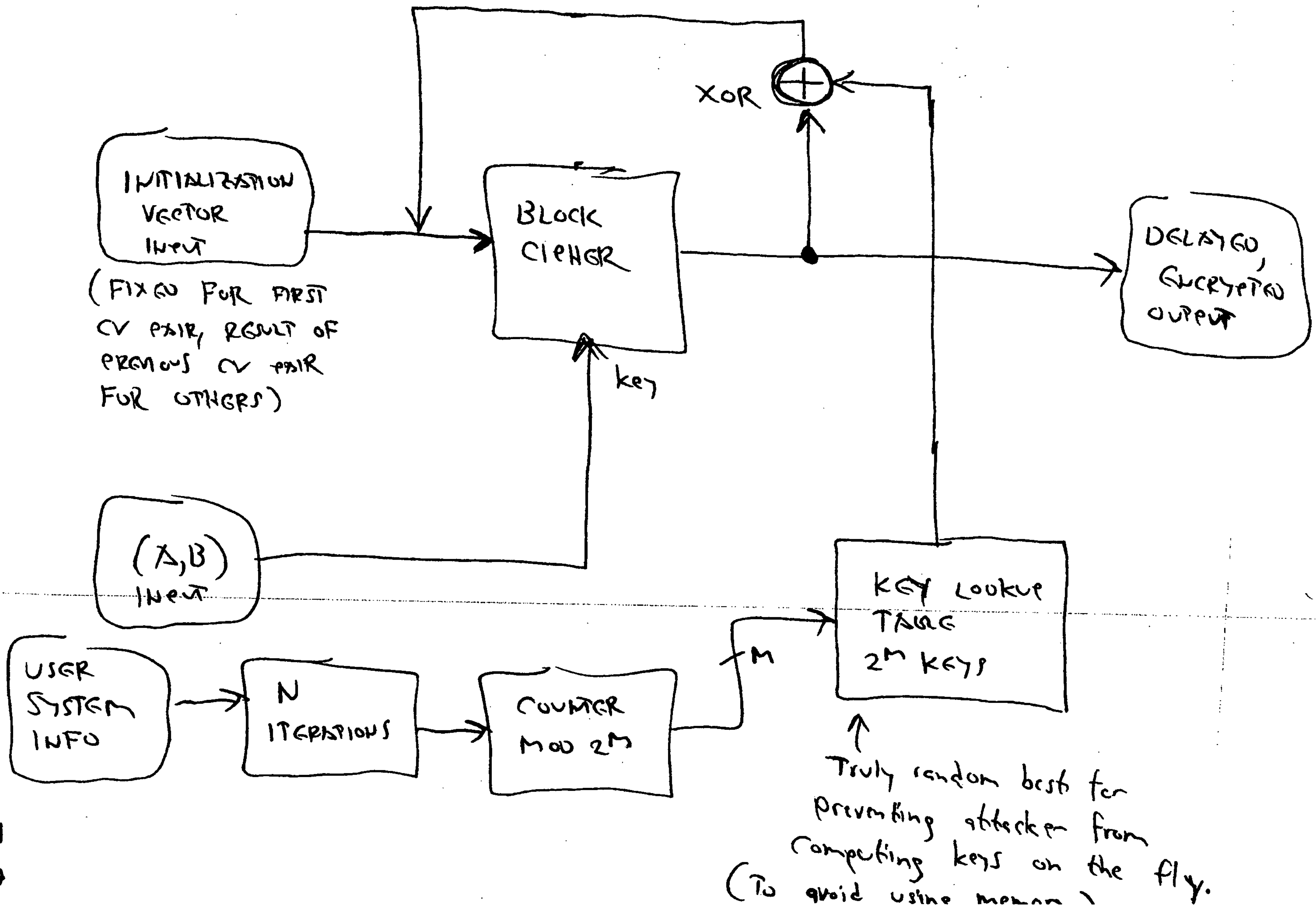
Z2-Z6 illustrate screen shots of a secure passphrase entry system according to various aspects of the invention, illustrating an exemplary user interface at different points during the input of a passphrase without the use of keystrokes. Thus, the security hazard of keystroke loggers can be avoided. In addition, the mouse-based input method may be preferred by users over the use of a keyboard, for example when they are entering their passphrase to browse encrypted e-mails or files. In an experiment the applicant carried out, "entering" the passphrase by the mouse input method (simulated by tapping a pen onto a printout similar to Z2-Z6) did not take him much longer than typing in the passphrase.

Advantageously, the passphrase is represented in the illustrated embodiment (as it is entered) both as circled letters and has a pair of stair-stepped line segments having characteristic shapes. Viewing the passphrase and its associated characteristic shapes of the line segments helps the user to remember the passphrase. Human brains are good at remembering pronounceable words (even when they are nonsense words) and are also good at remembering characteristic shapes. The combination of both characteristics of a unique passphrase can be expected to improve the user's ability to remember it when the time comes to input the passphrase.

A delay system according to another aspect of the invention, illustrated in the block diagrams of Z7 and Z8, makes a secure delay according to various aspects of the invention less unobtrusive to the user. It does so by beginning the delay process when the passphrase has been partially entered. Advantageously, such a system performs the delay computations substantially in parallel with the unavoidable delay of the user's input of the passphrase. Even when typing quickly, it took the applicant at least about three seconds to enter the passphrase during his experiment. This is a substantial period of delay that, when made computationally unavoidable, makes cracking the 2^48 possible combinations of the randomly chosen passphrase nearly impossible with the computing horsepower available around the date of filing of the present application. (See Z9 and Z10 for a detailed computational analysis.) The screen shots of Z2-Z6 show the "private key delayed unlocking" beginning with the first consonant-vowel pair entered by the user. The delayed unlocking (the inventive "computationally unavoidable" delay) continues substantially in parallel with the user's input of additional consonant-vowel pairs. Note Z6, in which the passphrase is confirmed and the private key has been completely unlocked.

\# # #


PASSPHRASE ENTRY USER INTERFACE
8 CV pairs over 3-10 seconds of user input
set CV: pair of consonant, vowel
13×5=65 members of set CV
TRANSFORM TO SET A: {0,1,... 64}
7
A
B
(A,B)
121
"FINGERPRINT" HASH OF USER'S PUBLIC KEY
Iterate 8 times
I.V.
SECURE ENCRYPTION AND DELAY
Using inventive indexed key lookup, # cycles selected to give ~1/3 sec. Delay on user's machine
1/3 × 8 sec ~2.5 sec not very noticeable
Could be public in view of passphrase + secure delay security
ENCRYPTED "LOCKED" PRIVATE KEY
publicly available
USER'S PUBLIC KEY
PUBLIC KEY / PRIVATE KEY PAIR GENERATOR
once
many
USER'S PRIVATE KEY
once
many
K
SYMMETRIC ENCRYPT.
very sensitive, never stored anywhere



INITIALIZATION VECTOR INPUT
(FIXED FOR FIRST CV PAIR, RESULT OF PREVIOUS CV PAIR FOR OTHERS)
BLOCK CIPHER
key
XOR
DELAYED, ENCRYPTED OUTPUT
(A,B) INPUT
USER SYSTEM INFO
N ITERATIONS
COUNTER MOD 2M
M
KEY LOOKUP TABLE 2M KEYS
Truly random best for preventing attacker from computing keys on the fly.
(To avoid using memory)

1/2

Number of consonants	13	
Number of vowels	5	
Combinations in each CV pair	65	
Pairs	8	
Total Combinations	**318,644,812,890,625**	
Base-2 Entropy	48	(bits)

Mean Input Times (experiment)

Touch typing (fast), hidden digits	4.70 (sec.)	=(5.3+ 4+ 5+ 5.1+ 4.1)/5
Tough typing (fast), digits shown	3.88	=(3.8+ 3.5+ 3.4+ 3.5+ 5.2)/5
Mouse, drag line through digits	9.32	= (9.7+ 9.2+ 9.8+ 8.9+ 9)/5
Mouse, click on digits	8.28	= (8+ 8.2+ 8.9+ 8+ 8.3)/5

Set total delay to minimum total input time (Keeps user from noticing the delay)	3.88 (sec.)

Software (equivalent machine) Attack Analysis

Total number of seconds for all delayed combinations (on equivalent machine)	1,236,341,874,015,620	
Average number of years on equivalent machine (1/2 total)	19,602,072	Impossible with present machines
Effective lifetime of signing key (years)	20	
Performance multiplier at end of life (Moore's law)	10,321	
Total number of seconds for all delayed combinations (on future machine)	119,785,790,491	
Number of future machines in network	1,000	Even this number would leave evidence of fraudulent activity on the part of the person forging signature with broken private key
Average number of years on future machine network (1/2 total)	**0.95**	

Massively Parallel Hardware (FPGA, ASIC) Attack Analysis

Budget (current equivalent dollars)	1,000,000	
Cost per FPGA or ASIC (with NRE)	400	
Number of available parallel branches in budget	2,500	
Number of parallel branches operating simulateously	2,048	
Performance multiplier of each branch over equivalent machine	100	
Total performance multiplier over equivalent machine	204,800	
Total number of seconds for all delayed combinations	6,036,825,557	
Average number of years (1/2 total)	96	Impractical with present H/W, even w/ large budget
Effective lifetime of signing key (years)	20	

Can Include sunset date in ACI after which all sigs. are invalid

Performance multiplier at end of life (Moore's law)	10,321
Total number of seconds for all delayed combinations (on future hardware system)	584,892
Average number of days on future machine (1/2 total)	**3.38**
But, here's where the key lookup helps protect against such attacks...	
Random keys in key lookup table	8,192
Size of each key (in bytes)	16
Total memory for lookup table (bytes)	131,072
Total fast SRAM memory for all branches (bytes)	268,435,456
Total MB of fast SRAM memory	**262,144**
Cost per MB of SRAM (current equivalent dollars)	10
Total cost of SRAM (See budget above.)	**2,621,440**

(~~must~~ Should fit in 256K cache) to ensure top performance in user's machine. otherwise ratio between

$$\left[\frac{\text{Attacker Delay}}{\text{User Delay}}\right]$$

is reduced.

Lots of gates: $$

~ 8 × 2 gates per byte, or ~ 4.3×10^9 gates or ~2M gates per branch

Expensive ASIC!